AMENDMENTS TO DRAWINGS

See attached amended drawings.

AMENDMENTS TO THE SPECIFICATION

Please amend the paragraph beginning on page 7, line 17 with the following paragraph:

Referring to FIG. 1, there is a block diagram that illustrates aspects of a Redundant Array of Independent Disks (RAID) data storage system 100, according to one embodiment of the invention. System 100 includes computer 102 connected across I/O interface 104 to controller 106. Computer 102 is any type of computer having a processor (not shown) coupled to a memory (not shown) for fetching data and executing computer program instructions stored in the memory. Such computer program instructions are used to perform data I/O operations with respect to one or more disk drives 112-1 through [112-N] 112-K I/O interface 104 can be any type of I/O interface, for example, a Small Computer Systems Interface (SCSI) I/O bus, a Fibre Channel bus, or other type of bus (e.g. Ethernet, IPI, HIPPI, Fire-Wire, USB, or IDE). SCSI I/O protocols are known in the art of computers and computer programming.

Please amend the paragraph beginning on page 7, line 27 with the following paragraph:

Controller 106 is a disk array controller for striping data to and from disk drives 112-1 through [[112-N]] 112-K according to the particular RAID level being used in system 100. Various RAID levels are known in the art of configuring RAID data storage systems. Controller 106 includes a processor (not shown) for fetching data and computer program instructions stored in memory 107 to perform data consistency check operations 108, and to coordinate RAID operations across disk drives 112-1 through [[112-N]] 112-K. Memory 107 can be any combination of a fast memory, such as a semiconductor random access memory (RAM), a fast non-volatile memory, such as read-only memory (ROM) for an erasable read-only memory EPROM, and a slow magnetic memory, such as a hard disk.

Please amend the paragraph beginning on page 8, line 15 with the following paragraph:

Controller 106 is connected across I/O interface 110 to disk drives 112-1 through [[112-N]] 112-K. I/O interface 110 can be any type of I/O interface, for example, a fibre channel interface or a SCSI interface. Disk drives 112-1 through [[112-N]] 112-K represent any non-volatile, randomly accessible, rewritable mass storage device which has the ability of detecting its own storage failures. It includes both rotating magnetic and optical disks and

solid-state disks, for non-volatile electronic storage elements, for example, PROMS, EPROMs, and EEPROMS.

Please amend the paragraph beginning on page 8, line 26 with the following paragraph:

The inventive data consistency check applies over the sector stripes, and by extension, over the entire data stripe. Multiple sectors make up a data stripe. Referring to FIG. 2, there is a block diagram that illustrates structure and organizational aspects of an exemplary data stripe 200 spanning a plurality of disk drives 112 (e.g. 112-1, 112-2, ..., [[112-N]] 112-K) according to one embodiment of the invention. Stripe 200 comprises is a physical block of data written to each of the ~~N~~ K data storage devices 112-1 through [[112-N]] 112-K. A sector stripe 207 (e.g. 207-1, 207-2, ..., 207-K) comprises one parity sector and one data sector from each stripe. There is a parity sector and a header for each data sector and header in the sector stripe. The first physical block on each data storage device 112 are collectively considered to be the first data stripe [[200-1]] 210-1. The second physical block on each disk drive or other data storage device 112 are collectively considered to be the second data stripe [[200-2]] 210-2, and so on, for each subsequent physical block on the data storage devices 112-1 through [[112-N]] 112-K.

Please amend the paragraph beginning on page 9, line 9 with the following paragraph:

First Data Stripe [[200-1]] 210-1 includes a parity sector for storing data to re-generate host or user data stored in each data sector 208-1 through 208-N in data stripe 200. Each sector strip 207-1 through 207-K includes one of more data sectors 208-1 through 208-N. The number of data sectors 208-1 through 208-N in a sector strip 207-1 through 207-K depends on how the RAID has been configured. For example, a RAID configuration based on 8K sector strips 207-1 through 207-K results in 16 data sectors 208-1 through [[08-N]] 208-N being included in each sector strip 207-1 through 207-N. Each respective sector strip 207-1 through 207-K is stored on a respective disk drive 112-1 through [[N]] 112-K. For example, sector strip 207-1 is stored on disk drive 112, sector strip 207-2 is stored on disk drive 112-2, and the like.

Please amend the paragraph beginning on page 9, line 18 with the following paragraph:

As discussed above, traditional data consistency checking techniques are limited, both because they typically cannot determine if the user data stored in one or more data sectors 208-1 through 208-N is corrupt, and because they typically operate on a burdensome amount

of data to perform such data consistency checks. The invention provides a solution to these prior art limitations by providing several of data check codes (DCC) metadata objects stored in parity header 202 and data headers [[208-1]] 206-1 through [[208-N]] 206-K. These data check codes and there usage are described in greater detail relative to FIG. 3 and FIG. 4.

Please amend the paragraph beginning on page 9, line 25 with the following paragraph:

Each data sector 208-1 through 208-N, includes a respective data header 206-1 through [[206-J]] 206-K, as a result of extended sector formatting. Each respective data header 206-1 through 206-K includes a data check code sub-data sector (DSS_{ds}) metadata object derived from the user data stored in the respective data sectors 208-1 through 208-K. The one or more DSS_{ds} values can be Longitudinal Redundancy Check (LRC) values, Cyclical Redundancy Check (CRC) values or Checksum Values. The DCC_{ds} can include an arbitrary number of bytes, the number of bytes depending on the formula or procedure selected for generation of the DCC_{ds}. In a preferred embodiment, the one or more DSS_{ds} values are LRC values. In a preferred embodiment, each respective DCC_{ds} consists of two bytes of data. Parity sector [[204]] 208 includes parity header 202 as a result of extended sector formatting. As is illustrated, data stripe 200 includes one parity sector [[204]] 208 for each data sector in the stripe. Parity header 202 includes a data check code sub-parity sector (DCC_{ps}) value derived from the parity data stored in the parity sector [[204]] 208. The DSS_{ps} value can be Longitudinal Redundancy Check (LRC) values, Cyclical Redundancy Check (CRC) values or Checksum Values. In a preferred embodiment, the DSS_{ps} value is an LRC value. In a preferred embodiments, the DCC_{ps} consists of two bytes of data.

Please amend the paragraph beginning on page 13, line 26 with the following paragraph:

If there are no errors for this sector stripe, then the procedure replaces the stored DCCss with the calculated DCCss (340) and returns to process any more sector strips that may be present (307) in the manner described herein above. Otherwise, if there are errors in this sector stripe, the source of the error is determined before further action is taken (325). More particularly, a test is performed to determine if the error is with either a sector DCCds or with a sector DCCps. If the error is with one of the DCCds or the DCCps, then the number of errors becomes important. If there is one error with the DCCps and DCCds, the error may be correctable, however if there is more than one error (326) with the DCCps and DCCds, an uncorrectable error is optionally but preferably logged [[(327)]] (342) and the procedure

again tests of makes a comparison to determine if there are any more sector stripes to process (307) as already described herein above.

Please amend the paragraph beginning on page 14, line 8 with the following paragraph:

When there is only one error in either the DCCps or DCCds (326), the location of the error may be determined. If the error is in the data sector (327), then the data is regenerated from the remaining data and parity (328) and a new DCCds is generated (329). This is followed by generating a new DCCss (330) and correction of the error log(s) (333). If one the other hand, the error determined (327) is not in the data sector or the error is not an error with a sector DCCds or DCCps (325), then a new parity data is generated from user data (331) followed by generation of a new DCCps (332), generation of a new DCCss (330) and correction of the error log to reflect the status (333). Independent of whether the error was correctable (333) or uncorrectable [[(327)]] (342), the procedure returns to determine whether there are any more sector stripes to process (307) and repeats the steps and procedures already described until all sector stripes have been processed and the procedure ends (311).